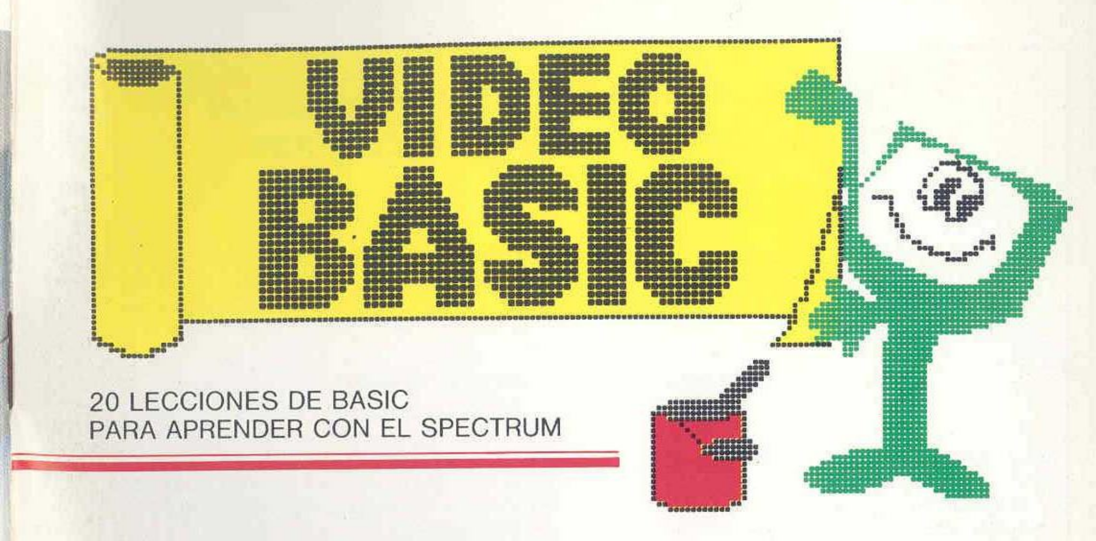


20 LECCIONES DE BASIC
PARA APRENDER CON EL SPECTRUM

INGELEK



JACKSON

- Qué es un robot
- Cómo enseñar a un robot
- El futuro de la robótica
- Proyecto y codificación de un carácter
- READ DATA
- RESTORE
- USR BIN
- Asignación de datos
- Videojuego N.º 11

11 Spectrum

16K/48K/PLUS

RESULTADO DEL SORTEO DE PROMOCION ENTRE LOS LECTORES DE VIDEOBASIC

El día 31 de julio, tal como establecen las bases del concurso VIDEOBASIC ante D. Juan M. BOLAS ALFONSO, notario del Ilustre Colegio de Notarios de Madrid, se ha efectuado el sorteo de 40 IMPRESORAS SEIKOSHA, cuyo resultado ha sido el siguiente:

SEIKOSHA GP 500 y GP 50* (LECTORES)

- ANTONIO ESPAÑA CERRATO. MADRID.
- VICENTE RODRIGUEZ CASTRO. PONFERRADA (LEON)
- JUAN JOSE FERNANDEZ COLLADO. CADIZ.
- JOSE JAVIER GOMEZ FERNANDEZ. MADRID.
- JAIME MAYORAL FERRER. SAN FELIU DE LLOBREGAT (BARCELONA).
- JOSE MARIA FERNANDEZ BENEYTO. BARCELONA.
- JOSEP FARREGAS SALA. VALLIRAMA (BARCELONA).
- FRANCISCO JOSE TAMARGO GARCIA.
- SALINAS-CASTRILLON (ASTURIAS).
- FELIPE SOTO GONZALEZ. SAN SEBASTIAN
- (GUIPUZCOA).
- EDUARDO MORENO MACIAS. SEVILLA.
- SALVADOR BLANQUER AZWAR. VALENCIA.
- RICARDO IBAÑEZ PUYAL. HOSPITALET (BARCELONA).
- SILVESTRE MATEOS REDONDO. SALAMANCA.
- SOFIA MORALES GARRIDO. MADRID.
- MANUEL GUILLEM TECLA. VALENCIA.
- GEMA COTEREAU BADIOLA. MADRID.
- JAIME SANCHEZ VAZQUEZ. SEVILLA.
- MAN CHUNG LAM. FUENGIROLA (MALAGA).
- MANUEL MANZANO MARTINEZ. VALENCIA.
- CESAR RODA MARTINEZ. REQUENA (VALENCIA).

*** Los agraciados deberán enviarnos fotocopia de la tarjeta de garantía de su ordenador para remitirles la impresora adecuada.**

SEIKOSHA GP 500 (SUSCRIPTORES)

- JOSE A. CAMARGO MORALES. MADRID.
- ANGEL MARTIN AGUADERO. BEJAR (SALAMANCA).
- ISABEL MEDIAVILLA SANCHEZ. SANTANDER.
- MANUEL CARMONA LUQUE. UTRERA (SEVILLA).
- REGINA RAMOS GOMEZ. BARCELONA.
- MIGUEL ANGEL ALONSO JIMENEZ. MADRID.
- ROBERTO PECINO CANO. LA LINEA (CADIZ).
- FERNANDO ANGOSO SANCHEZ. SALAMANCA.
- MARIANO RUMBERO SANCHEZ. VALENCIA.
- ANTONIO CRESPO INSAUSTI. LAS ARENAS (VIZCAYA).

SEIKOSHA GP 50 (SUSCRIPTORES)

- TOMAS ROMERA SANZ. MADRID.
- FRANCISCO GUTIERREZ FERNANDEZ. MADRID.
- EDY SERI. MADRID.
- JUAN PABLO VERDU MIRA. SAN VICENTE RASPEIG (ALICANTE).
- ALBERTO SOLE BAQUES. SITGES (BARCELONA).
- FRANCISCO VAZQUEZ LAZARO. BILBAO (VIZCAYA).
- IGNACIO JAVIER PONTE MARTIN. AGUADULCE (ALMERIA).
- JORGE GRAU CRESPO. CASTELLO DE RUGAT (VALENCIA).
- JOSE MARIA NEBOT GOMEZ DE SALAZAR. MADRID.
- JOSE MANUEL AVILA FERNANDEZ. LEGANES (MADRID).

VIDEO BASIC
Una publicación de
INGELEK JACKSON
Director editor por INGELEK:
Antonio M. Ferrer
Director editor por JACKSON HISPANIA:
Lorenzo Bertagnolio
Director de producción:
Vicente Robles
Autor: Softidea
Redacción software italiano:
Francesço Franceschini,
Stefano Cremonesi
Redacción software castellano:
Fernando López, Antonio Carvajal,
Alberto Caffarato, Pilar Manzanera
Diseño gráfico:
Studio Nuovaidea
Ilustraciones:
Cinzia Ferrari, Silvano Scolari,
Equipo Galata
Ediciones INGELEK, S. A.
Dirección, redacción y administración,
números atrasados y suscripciones:
Avda. Alfonso XIII, 141
28016 Madrid. Tel. 2505820
Fotocomposición: Espacio y Punto, S. A.
Imprime: Gráficas Reunidas, S. A.

ISBN del tomo 3: 84-85831-19-5
ISBN del fascículo: 84-85831-11-X
ISBN de la obra completa: 84-85831-10-1
Depósito Legal: M-15076-1985
Plan general de la obra:
20 fascículos y 20 casetes, de aparición quincenal,
coleccionables en 5 estuches.
Distribución en España:
COEDIS, S. A.
Valencia, 245. 08007 Barcelona.
INGELEK JACKSON garantiza la publicación de todos los fascículos y casetes que componen esta obra y el suministro de cualquier número atrasado o estuche mientras dure la publicación y hasta un año después de terminada.
El editor se reserva el derecho de modificar el precio de venta del fascículo, en el transcurso de la obra, si las circunstancias del mercado así lo exigen.
Agosto, 1985
Impreso en España.

INGELEK



JACKSON

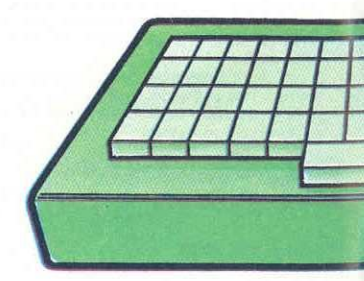
SUMARIO

Introducción

En esta lección veremos en primer lugar cómo funcionan, cómo aprenden, y cómo trabajan los robots, comentando también las perspectivas fascinantes y las posibilidades que estos dispositivos pueden ofrecer para la futura vida cotidiana.
Contemplaremos a continuación cómo es posible insertar permanentemente en un programa informaciones útiles o recurrentes, usando las instrucciones READ, DATA y RESTORE.
Y para terminar veremos un ejemplo sobre la técnica a adoptar para personalizar la salida de visualización, aprendiendo a definir y memorizar nuevos caracteres en el interior de nuestro ordenador.

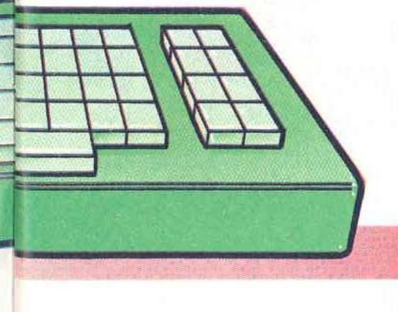
Qué es un robot

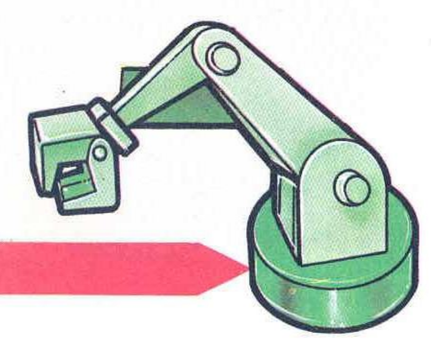
Ya desde antes de la llegada de los ordenadores electrónicos, una de las mayores aspiraciones del hombre fue la de conseguir inventar y construir nuevas máquinas que fueran capaces de relevarle en los trabajos más pesados, peligrosos o repetitivos.
La historia siempre nos lo enseña: cada peldaño de la escala del progreso ha estado normalmente caracterizado (y así sigue siendo hoy en día) por un gran descubrimiento o invención fundamentales (el fuego, el hierro, el vapor o la electricidad), descubrimientos que habiendo sido inicialmente realizados para resolver determinados problemas concretos, al ser aplicados y desarrollados nos han proporcionado, y siguen proporcionándonos, nuevos perfeccionamientos y progresos a veces inesperados.
El ordenador electrónico, aunque no pueda considerarse una invención en sentido estricto, no escapa a esta regla.
Nacido para «calcular», el ordenador se ha convertido en la base de una infinidad de nuevas aplicaciones, que se basan y dependen estrictamente de él. Entre estas encontramos a la robótica, es decir, la ciencia que se ocupa de la automatización del trabajo mediante máquinas controladas por ordenador.
Robot es un término que suscita en la mente de muchas personas dudas y perplejidades: el recuerdo de hombrecillos metálicos y de máquinas parlantes, tan querido por la ciencia-ficción comercial, hacen quizá que sea difícil la comprensión exacta del tema; así que en primer lugar intentaremos explicar su significado exacto.
Un robot no es más que una máquina automática, que bajo el control de un ordenador, desarrolla un trabajo preestablecido: por ejemplo, barnizar un automóvil, apretar tuercas o soldar chapas.



HARDWARE

La gran ventaja de un robot sobre una máquina automática normal es que al estar conectada a un ordenador, puede ser programado —con modificaciones que normalmente no resultan muy caras— para desarrollar distintos trabajos (aunque no demasiado diferentes).
A la programabilidad hay que añadirle aún otra ventaja: gracias a los progresos de la microelectrónica, los robots están dotados de unos dispositivos especiales llamados sensores que —aún estando francamente alejados de los sentidos humanos— les permiten reconocer cuándo ocurren determinadas situaciones especiales o anómalas.
Veamos un ejemplo. Supongamos que tenemos que unir con una tuerca dos piezas de metal adecuadamente preparadas. Si usamos un robot para esta operación, nos podemos cuidar de la aparición de determinadas eventualidades, debidas a un mal alineamiento de las piezas o al tamaño inadecuado de uno cualquiera de los componentes. Proporcionándole a la máquina los sensores adecuados y programando debidamente el ordenador de control, es posible parar la máquina cuando ocurrra uno cualquiera de estos casos. Una máquina normal, en presencia de estas situaciones anómalas seguiría con su trabajo, provocando la puesta en producción de una pieza defectuosa, o aún peor, dañando a la máquina misma.
Así, la introducción del robot le añade a la cadena de montaje un elemento de control y comprobación, además de productivo. Además, muchos robots pueden desempeñar trabajos que serían peligrosos o desagradables para las personas, como mediciones de radioactividad en una central nuclear, desactivar una bomba o trabajar en atmósferas contaminadas.

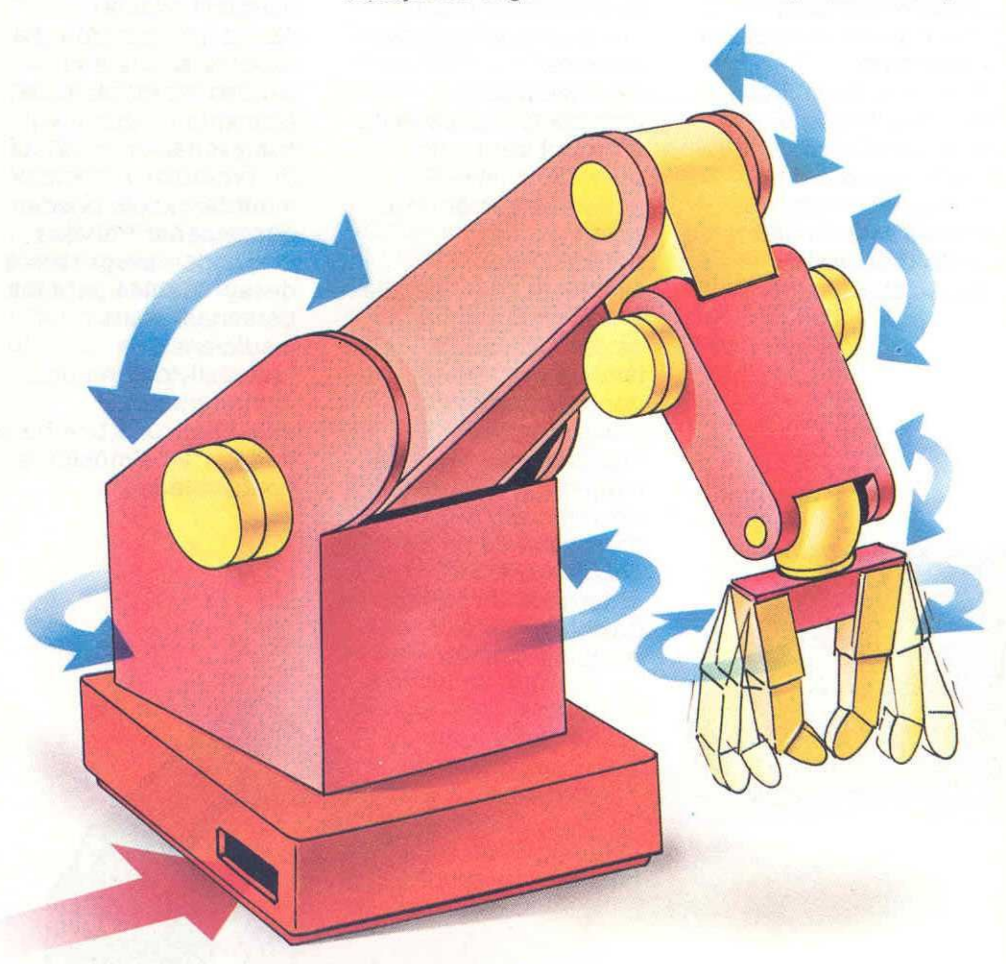




Cómo enseñar a un robot

La mayor parte de los robots existentes trabajan en fábricas en las que todo está organizado a su alrededor: normalmente, tienen que tomar piezas de las cintas transportadoras, trabajarlas según una cierta sucesión de operaciones elementales y, finalmente, depositarlas sobre otra cinta transportadora, que se ocupa de enviarlas hacia otros puntos de elaboración.

De cualquier manera,

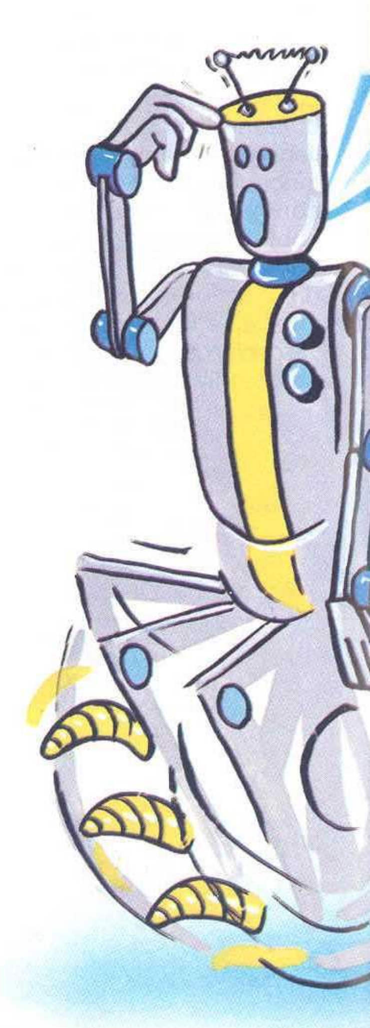


las prestaciones de un robot dependen de muchos factores, y no es el último el económico; por lo tanto, existen (y estos son la mayoría, dado su menor coste y su relativa sencillez) robots construidos para usos bastante generales, que son adaptados en cada caso para desarrollar tareas más específicas.
Los robots más comunes son los llamados «de brazo»: su mecánica está construida tomando como modelo el brazo humano. Esta disposición permite —al igual que en los verdaderos brazos— una notable libertad y capacidad de movimientos, permitiéndole a las «manos» (que suelen ser pinzas, conectadas a motores eléctricos) trabajar en posiciones que de otra forma serían inaccesibles.
Actualmente, se está estudiando también la forma de dotar a estas «manos» de una especie de sentido del tacto, permitiendo así una tracción adecuada al peso y a la solidez del objeto a levantar (hasta ahora ningún robot consigue distinguir una pieza de hierro con la forma de un huevo de un huevo verdadero), con resultados fácilmente comprensibles.
Por lo tanto, proyectar, construir y enseñar a un robot es una operación nada sencilla: se necesita en primer lugar analizar y subdividir la acción que el robot tendrá que desarrollar en todos sus posibles e imaginables pasos elementales (¿te acuerdas de los algoritmos?...), clasificando así todas sus diferentes necesidades de movimiento. En base a estos movimientos —y en función del esfuerzo necesario para realizar cada una de estas acciones— se decidirá el número y la posición de los brazos mecánicos, de las juntas (es decir, de las articulaciones entre los distintos brazos) y de sus motores respectivos.
Además, para poder realizar su trabajo, el robot tendrá que estar dotado de varios tipos de sensores (fotoeléctricos, electrónicos o mecánicos), adecuados para el tipo de trabajo requerido. A través de uno o más interfaces estos sensores tendrán que enviarle informaciones codificadas al ordenador de control, que supervisa la dirección y la coordinación de los movimientos, que así podrá dirigir la ejecución, la modificación o la parada de las operaciones previstas desde un principio.
Por lo tanto, el ordenador de control habrá de programarse adecuadamente para este tipo de necesidades, teniendo debidamente en cuenta todos los imprevistos y

anomalías posibles. Y esto se repite para decenas o centenares de acciones singulares. Una vez más habrás de darte cuenta de la importancia que tiene —antes de pasar a la verdadera fase de programación— la evaluación, hasta en los más pequeños detalles, de todo el conjunto de operaciones, y esto independientemente de que sean importantes o insignificantes, puesto que el microprocesador del robot —una vez que sea operativo— siempre tendrá que controlar la totalidad.
En un robot, y especialmente si es de tipo industrial, esta fase de análisis reviste una especial dificultad, dado que normalmente un único microprocesador no es capaz de enfrentar, él solo, la entera mole de trabajo que el robot suele poder desarrollar. Habitualmente, por no decir siempre, se hace necesario programar una serie de microprocesadores para que desarrollen todas las operaciones particulares (como por ejemplo, mover un brazo, o coger un objeto), coordinándolos a todos mediante una gran central de control capaz de seguir en cada momento las fases de elaboración.
Como podrás imaginar, si ya es difícil programar un solo ordenador, programar diez o quince (y para que encima puedan trabajar coordinados entre sí) se convierte en una empresa extremadamente delicada.

Desarrollos futuros

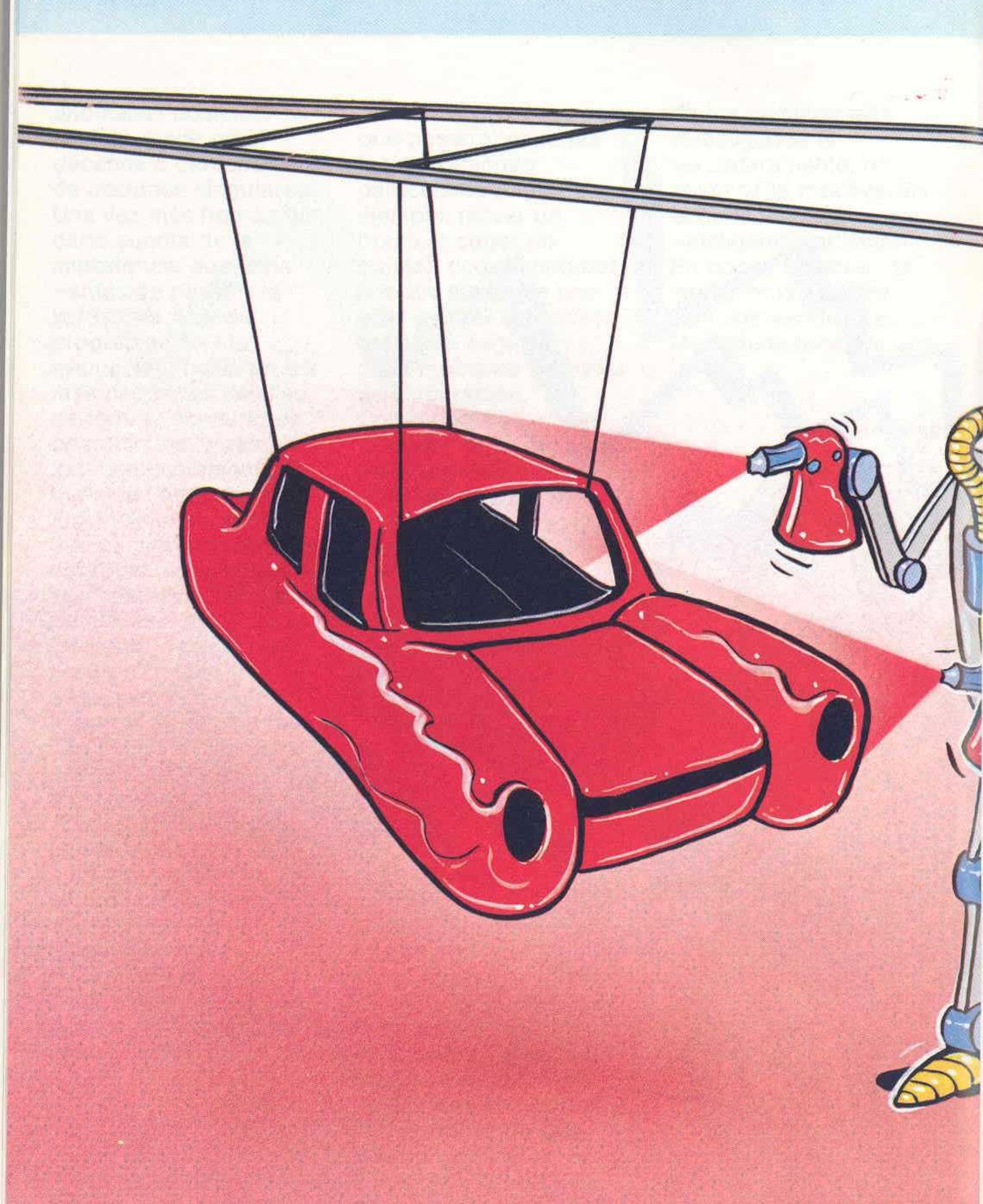
La robótica es una ciencia muy joven, se puede decir que apenas en sus principios. En el futuro podrá haber, por lo tanto, muchos desarrollos y mejoras, gracias también a los numerosos proyectos de investigación que están actualmente en curso en todo el mundo.
En este momento, uno de los caminos más investigados (y verdaderamente, no sólo por la robótica) es el de la llamada «inteligencia artificial». En pocas palabras, la inteligencia artificial está intentando dotar a los ordenadores de una

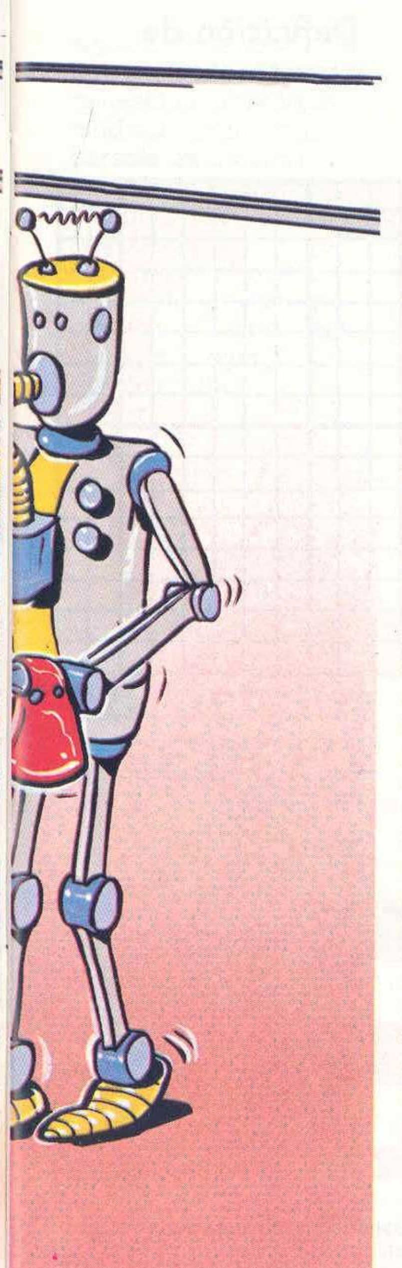


HARDWARE



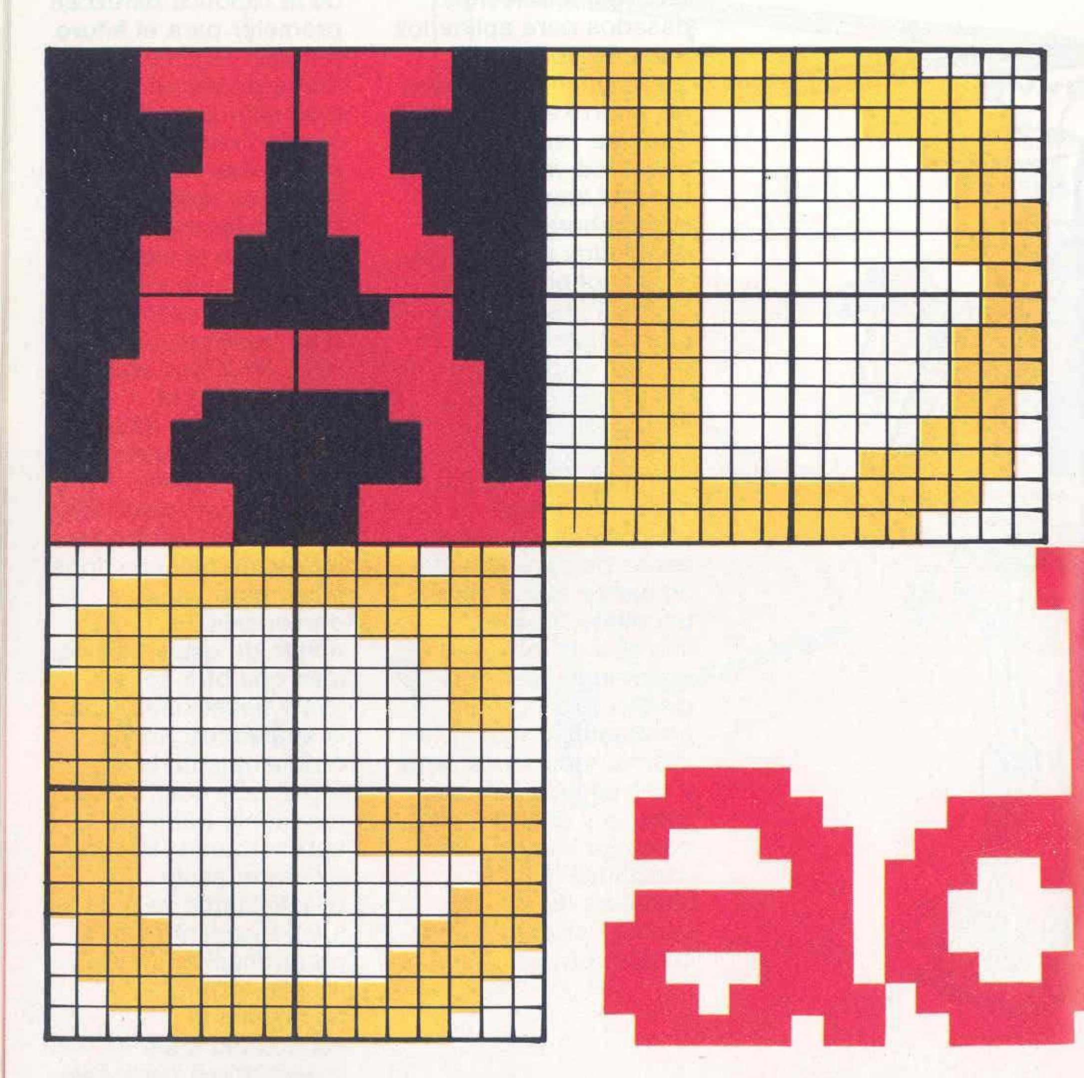
HARDWARE





especie de inteligencia autónoma y automática, capaz de aprender y recordar los hechos pasados para aplicarlos a los futuros (exactamente igual que les ocurre a los niños, que una vez se han quemado ya no intentan tocar el fuego).
Hasta ahora los resultados no han sido muy esperanzadores, pero da la sensación de que en pocos años se podrán empezar a ver las cosas más claras.
De cualquier forma, más allá de los temores que pueda inspirar la idea de máquinas inteligentes, un robot capaz de aprender del ambiente que lo rodea, permitiría resolver muchos de los problemas ligados al diseño y a la programación del robot mismo, evitando la larga y complicada fase de estudio y análisis, que como ya hemos visto constituye la mayor dificultad en la construcción e instalación de todos los dispositivos automáticos.
Aunque los desarrollos de la robótica parezcan prometer para el futuro próximo algunas cosas interesantes, en el momento actual aún quedan muchos problemas: los robots no saben moverse como desearíamos (nadie por el momento ha conseguido hacer andar a un robot como a un hombre), cuestan bastante caros (sobre todo los robots industriales) y no es posible adaptarlos para usos universales.
De cualquier modo, en el mercado existen ya robots de muchos tipos, hasta para usos personales. Pero el objeto de estos últimos, más que otra cosa, suele hacer referencia a la enseñanza de los rudimentos de la robótica, y sólo raramente permiten aprender con facilidad, y con un gasto relativamente bajo, el funcionamiento y la programación de los robots, siendo normalmente conectable a los ordenadores personales más difundidos, para poder controlar sus movimientos.

Definición de caracteres

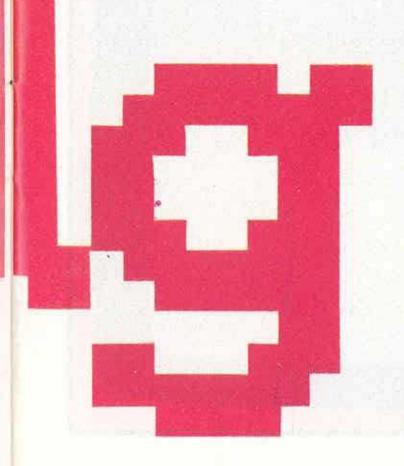


Un carácter está encerrado en una matriz de 8 por 8 puntos. Cuando desees representarlo en detalle, será necesario operar sobre una matriz de puntos mayor.
En el ejemplo puedes observar cómo para reproducir detalladamente un alfabeto especial se ha recurrido a una matriz de 16 por 16 (4 caracteres).

LENGUAJE

Nos ocurre a veces que, por gusto o por necesidad, deseamos modificar uno o más caracteres de entre aquellos normalmente disponibles en tu Spectrum.
Para hacer esto es necesario realizar una sencilla operación, más conocida como definición de caracteres, que permite crear un nuevo conjunto de símbolos que se hacen corresponder con cada una de las teclas existentes en el teclado.
Así, si por ejemplo deseáramos disponer en el teclado de las letras del alfabeto griego, sería suficiente con que le «enseñáramos» al ordenador la forma de cada uno de los nuevos caracteres, de forma que a la tecla «A» le corresponda la «alfa», a la letra «B» la «beta» y así sucesivamente.
Tú ya sabes que es lo que hace el Spectrum para ser capaz de visualizar todos los caracteres que normalmente puedes ver en pantalla, y sabes también que la forma de los caracteres (constituida por series de 8 bytes para cada carácter) se encuentran cuidadosamente guardadas en la memoria de sólo lectura (ROM), para que no se pierdan cada vez que se apaga el ordenador.
Por lo tanto, cuando pulsas una tecla, ésta se corresponde con una dirección de la ROM que contiene la secuencia de informaciones que el ordenador necesita conocer para que el carácter pueda aparecer en pantalla.
Dado que la memoria ROM no puede ser alterada de ninguna manera, lo primero que hay que hacer, si se desean definir nuevos caracteres, es transferir todas las imágenes de los antiguos caracteres a una zona accesible también a la escritura (es decir, a la memoria RAM).
Ahora ya está todo predispuesto para aceptar las eventuales modificaciones que deseemos aportar a los distintos caracteres.
Sin embargo, para definir un nuevo carácter es necesario examinar en primer lugar qué se debe de hacer para crear un carácter. Cada carácter está compuesto por una combinación de 64 puntos (obtenida

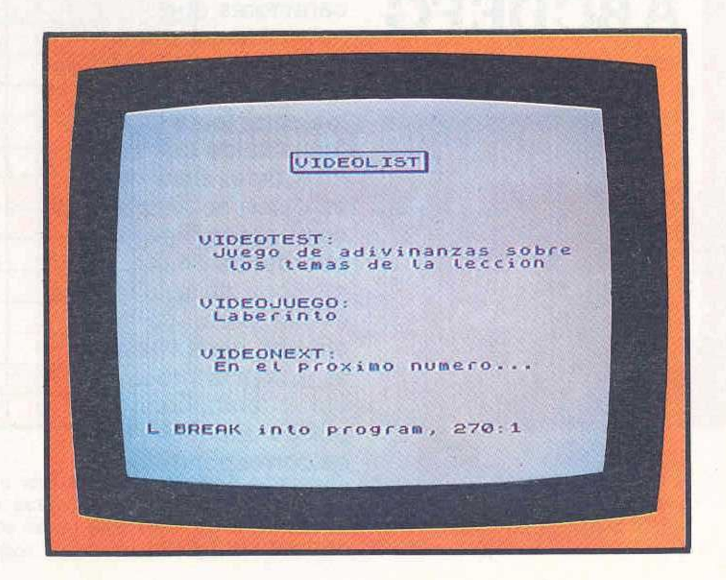
abcdefghij
ABCDEFG



empleando los 8 bits de cada byte) dispuestos sobre 8 líneas y 8 columnas. He aquí un ejemplo, representado por una secuencia de este tipo:

BINARIO	DEC.
0 0 1 1 1 1 0 0	60
0 1 1 0 0 1 1 0	102
0 1 1 0 1 1 1 0	110
0 1 1 0 1 1 1 0	110
0 1 1 0 0 0 0 0	96
0 1 1 0 0 1 1 0	102
0 0 1 1 1 1 0 0	60
0 0 0 0 0 0 0 0	0

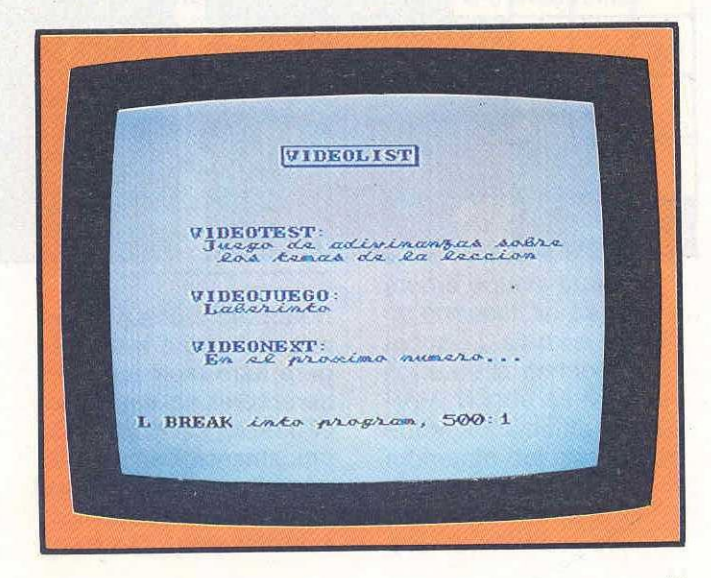
donde cada 1 ó 0 representan respectivamente un punto (pixel) encendido o apagado.
Para un ordenador, encendido o apagado significan valor 1 ó 0: entonces será suficiente con introducir en las localizaciones dedicadas a la definición de ese carácter los ocho números binarios, leídos línea por línea, que definen las combinaciones de pixels.
En nuestro ejemplo, tomando la primera línea (00111100) obtenemos un determinado número

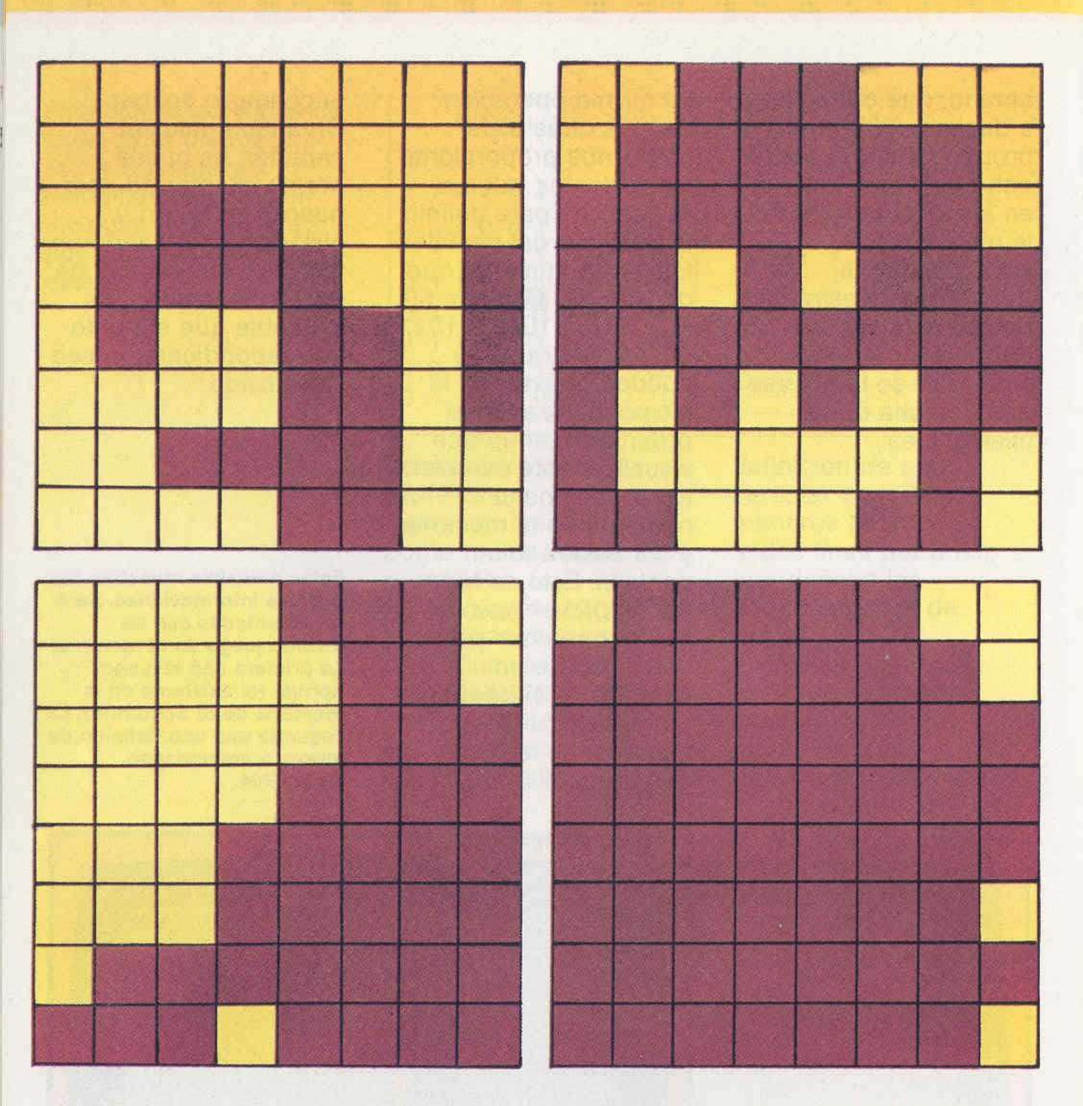


binario, que convertido a decimal (60), nos proporciona uno de los ocho valores a insertar en las localizaciones de la memoria que especificarán la descripción y estructura de ese carácter.
Para mayor claridad, este valor se ha escrito a la derecha de la misma línea.

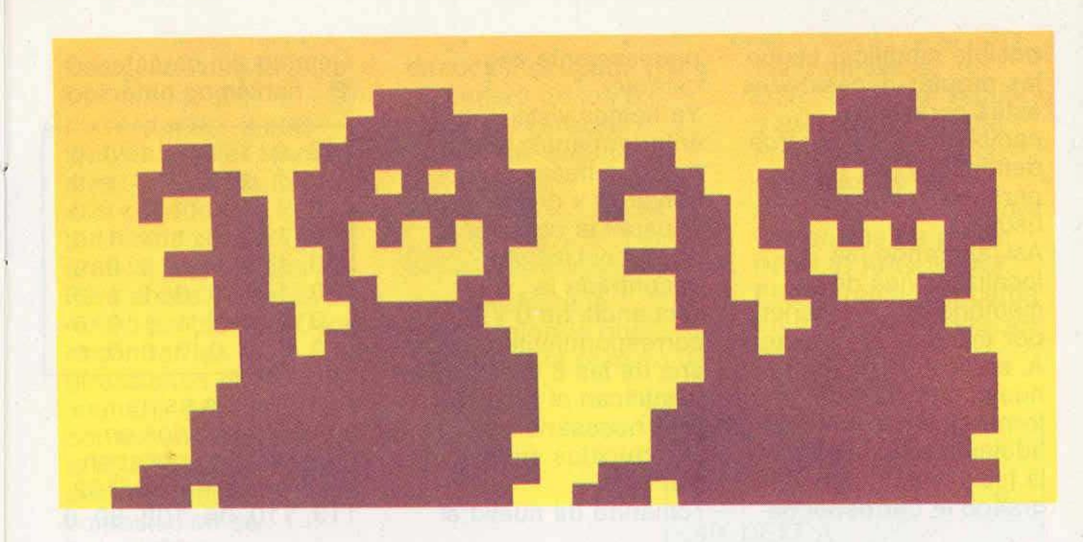
La misma operación con las otras siete líneas nos proporciona los restantes valores, necesarios para definir la totalidad del carácter. Los ocho números que derivan de C serán: 60, 102, 110, 110, 96, 102, 60 y 0. Ahora ya se pueden insertar en la memoria. Cuando el ordenador tenga que visualizar este carácter, tomará los ocho números de la memoria y los pasará a la pantalla. Esto es todo. Así, el primer paso que hay que realizar para la definición de un carácter es el de dibujar una cuadrícula de 64 cuadros, en la que definir los pixels a encender o apagar. Creando cualquier carácter, es buena norma intentar no usar nunca un 1 o un 0 aislados: empleando un televisor que no sea de excelente calidad, es probable que el punto correspondiente no sea visualizado.

Estas pantallas muestran las mismas informaciones, pero representadas con un distinto juego de caracteres. La primera usa el juego normal (el existente en la memoria de tu Spectrum). La segunda usa uno definido de nuevo, y memorizado en la RAM.





Hecho esto una primera vez, el camino a seguir para introducir otros caracteres será siempre el mismo: cambiarán únicamente los números a insertar en el ordenador (cada carácter está claramente definido por una particular y única secuencia de puntos luminosos) y las direcciones y localizaciones en las que introducir estos números. Al final, el conjunto de caracteres



habrá sido definido y podrá ser empleado a placer en cualquier programa.
Para recuperar el antiguo conjunto de caracteres será suficiente con apagar el ordenador, volviéndolo a encender después de algunos segundos: ésta operación borrará todas las modificaciones efectuadas en la memoria RAM, reestableciendo las direcciones de la ROM que le corresponden a las teclas habituales.

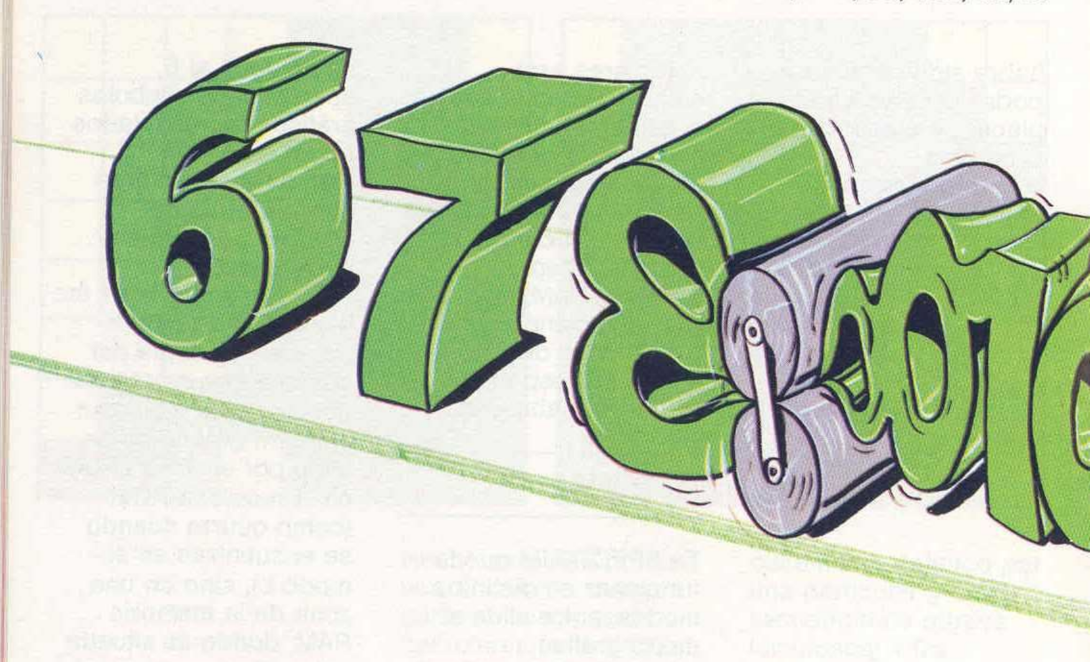
USR-BIN

Tu SPECTRUM puede funcionar en distintos modos, entre ellos el modo gráfico.
¿Qué significa gráfico? Es bien sencillo: cuando te encuentras en modo gráfico (cursor G), pulsando las cifras del 1 al 8, aparecen los símbolos gráficos representados sobre las teclas respectivas, mientras que las letras del alfabeto permanecen inalteradas.
Pero el asunto no es tan banal como pudiera parecer. La copia del alfabeto que se obtiene pulsando las teclas en el modo gráfico no es leído por el SPECTRUM en la memoria ROM (como ocurre cuando se encuentran en el modo L), sino en una zona de la memoria RAM, donde es situada automáticamente en el momento del encendido del ordenador.
Esto significa que, queriéndolo así, es

posible modificar según las propias necesidades estos caracteres, cambiándolos por otros definidos y personalizados por el usuario.
Así, alterando las localizaciones de memoria que contienen, por ejemplo, el carácter A, es posible definir un nuevo carácter, de forma que de allí en adelante a la presión de la tecla A en modo gráfico le corresponda precisamente ese carácter.
Ya hemos visto anteriormanete lo que hay que hacer para construir y definir sobre el papel el carácter deseado. Una vez encontrada la secuencia de 0 y 1 correspondiente a cada una de las 8 líneas que identifican el carácter, será necesario introducirlas en la memoria.
Tomando de nuevo el ejemplo del carácter 🄲, habíamos obtenido:

0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	0	1	1	1	0
0	1	1	0	1	1	1	0
0	1	1	0	0	0	0	0
0	1	1	0	0	1	1	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0

Convirtiendo estos números binarios en decimales se obtienen los 8 números: 60, 102, 110, 110, 96, 102, 60, 0.



LENGUAJE

Decidimos que la tecla que tendrá que corresponder a este nuevo carácter tendrá que ser la B, de forma que cuando la pulsemos en modo gráfico, se visualice ■. Pero ahora surge un nuevo problema: ¿Cómo encontrar las direcciones de la memoria RAM correspondientes a la B, en las que escribir estos 8 números? La contestación se llama USR.
USR es una función que convierte un argumento de tipo cadena compuesto por un único carácter en la dirección ocupada por el primero de los 8 bytes del carácter gráfico del argumento. Por lo tanto:

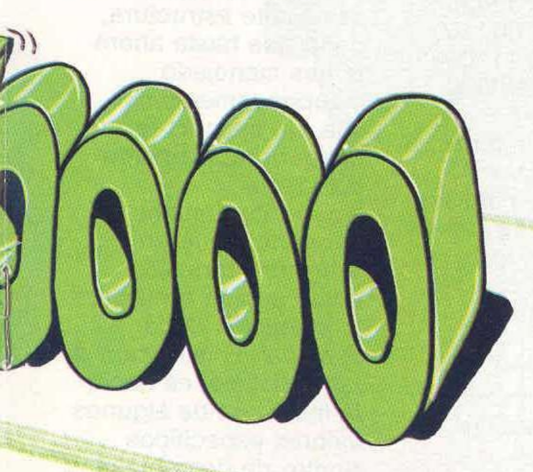
```
USR "B"
```

nos devolverá la dirección en la que memorizar el primero de los 8 números, USR "B" + 1 el segundo y así sucesivamente. Proporcionándole a USR más de un argumento se obtendrá el mensaje de error: INVALID ARGUMENT.
El programa que carga en memoria el carácter gráfico que hemos definido será:

```
10 FOR A = 0 TO 7
20 READ B
30 POKE USR "B" + A, B
40 NEXT A
50 DATA 60, 102, 110, 110
60 DATA 96, 102, 60, 0
```

Una vez ejecutado el programa, la imagen del carácter ■ habrá sido insertada en el ordenador y podrá ser llamada pulsando en el modo G (gráfico) el carácter B. Deseándolo, también es posible saltarse la fase de conversión de los números binarios a decimales empleando una segunda función: BIN.
BIN se emplea para introducir en el SPECTRUM un número en su forma binaria en lugar de la decimal. Por lo tanto:

```
POKE 300, BIN 11111111
```



es del todo equivalente a

```
POKE 300, 255
```

y, en efecto

```
PRINT BIN 11111111
```

imprime como resultado 255 decimal.
Así, nuestro programa, también se podría haber escrito de esta manera:

```
 10 FOR A = 0 TO 7
 20 READ B
 30 POKE USR "B" + A, B
 40 NEXT A
 50 DATA BIN 00111100
 60 DATA BIN 01100110
 70 DATA BIN 01101110
 80 DATA BIN 01101110
 90 DATA BIN 01100000
100 DATA BIN 01100110
110 DATA BIN 00111100
120 DATA BIN 00000000
```

En la práctica la función BIN nos permite introducir en las líneas DATA la matriz de puntos de un carácter, igual exactamente a como la hayamos dibujado.

Sintaxis de la función USR

Una cadena de un único carácter comprendido entre A y U

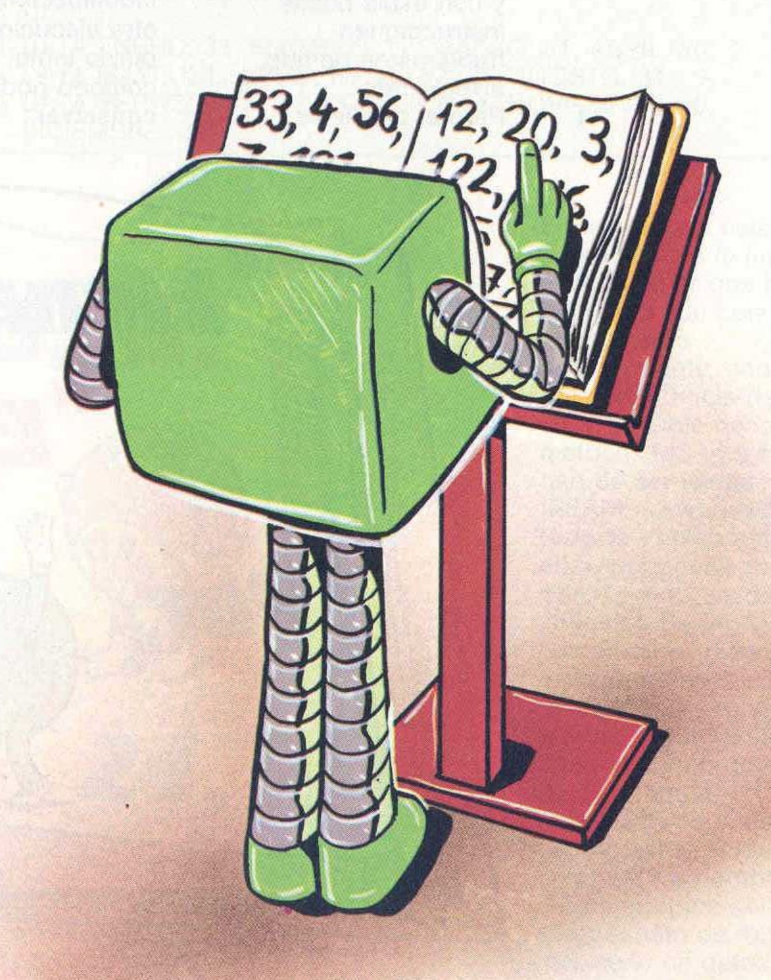
Sintaxis de la función BIN

BIN número binario con un máximo de 16 bits

READ/DATA

Las instrucciones READ y DATA permiten leer datos desde el interior de un programa, evitando así el tenerlos que indicar manualmente desde el teclado.
La introducción de los datos a elaborar se realiza además sin la momentánea parada del programa, contrariamente a lo que ocurre con la instrucción INPUT.
Con toda probabilidad te estarás preguntando para qué puede servir semejante estructura, dado que hasta ahora te has manejado estupendamente sin ella. Un ejemplo te aclarará rápidamente las ideas.
Ocurre con mucha frecuencia que un programa requiere como acción preliminar la ejecución de la llamada inicialización de las variables, es decir, la inserción de algunos valores específicos dentro de determinadas variables (por ejemplo, los nombres y la duración de cada mes del año).
Para realizar esta

operación tenemos a nuestra disposición tres posibles alternativas:
– emplear una larga serie de LET al principio del programa;
– pedir en cada instrucción, usando INPUT, los valores a asignarle a cada variable;
– emplear READ y DATA.
Descartamos rápidamente la primera solución: requeriría demasiado trabajo durante el tecleado del programa y ocuparía demasiada memoria (recuerda que cada instrucción conservada dentro del ordenador ocupa una cierta cantidad de memoria).
La segunda solución resulta ya más



aceptable. En el caso de los meses se podría escribir:

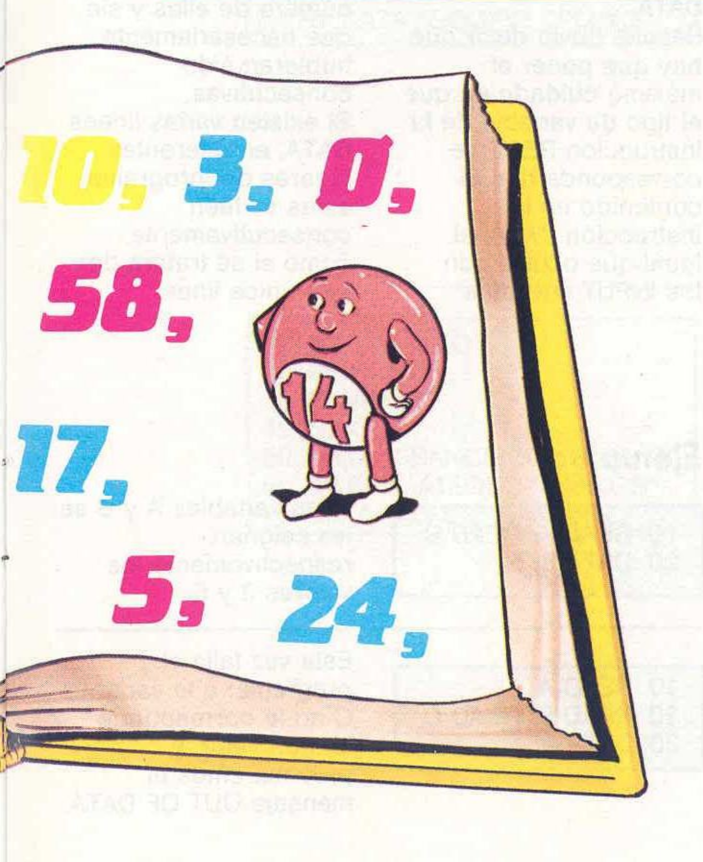
```
10 DIM M$ (12, 10) : DIM G (12)
20 FOR I = 1 TO 12
30 INPUT M$ (I), G (I)
40 NEXT I
```

y con estas pocas instrucciones hubiéramos podido arreglarnos. Pero el problema solamente ha quedado resuelto en parte: con cada RUN tendríamos que ponernos a la tarea de teclear ENERO, 31; FEBRERO, 28; etc., indicando toda una serie de valores, que, a fin de cuentas, no sufren ninguna modificación de una a otra ejecución, y que, por lo tanto, resultaría cómodo poder conservar



permanentemente dentro del programa. Esto significa que cada vez que empleemos este programa tenemos que asignar desde el teclado la totalidad de las 24 informaciones que nos son requeridas. Es un trabajo largo, aburrido e inútil. Adoptando READ y DATA, tendríamos en cambio:

```
10 DIM M$ (12, 10) : DIM G (12)
20 FOR I = 1 TO 12
30 READ M$ (I) : READ G (I)
40 NEXT I
50 DATA ENERO, 31, FEBRERO, 28, MARZO, 31, ABRIL, 30
60 DATA MAYO, 31, JUNIO, 30, JULIO, 31, AGOSTO, 31
70 DATA SEPTIEMBRE, 30, OCTUBRE, 31, NOVIEMBRE, 30,
   DICIEMBRE, 31
```



En la práctica estas instrucciones le indican al ordenador que haga lo mismo que con las líneas vistas anteriormente, con la única diferencia de que cada variable de las matrices M$ ( ) y D ( ) han de ser leídas (READ) no ya desde el teclado —como antes ocurría a causa de las INPUT —sino desde las líneas de DATA.
Los datos se insertan en el ordenador una única vez y, lo que es aún más importante, quien use el programa no se ve en la obligación de conocer y teclear estos datos (imagina los nombres de los equipos del campeonato de fútbol u otra serie de datos,

posiblemente menos conocidos que los nombres de los meses del año).
De esta forma es posible mantener, conservándolos en el interior del programa, valores que de otra forma se perderían todas las veces con el apagado del ordenador y con cada nueva ejecución del programa mismo.
Las instrucciones DATA contienen los distintos valores que se desean asignar; estas permiten reagrupar todos los datos en un único punto del programa, normalmente al principio o al final, donde resultan más fácilmente legibles.
Cada término de tipo cadena debe, como es norma, estar encerrado entre comillas.
La instrucción READ permite leer los datos que han sido memorizados en las DATA.
Resulta obvio decir que hay que poner el máximo cuidado en que el tipo de variable de la instrucción READ se corresponde con el contenido en la instrucción DATA, al igual que ocurre con los INPUT enviados desde el teclado: nunca deberá suceder que una variable numérica pueda contener un alfanumérico, o que una variable de tipo cadena se encuentre en un valor numérico. ¡Este tipo de error siempre está al acecho!
Las líneas DATA del listado anterior son tres únicamente por motivos de legibilidad: habríamos podido poner un distinto número de ellas y sin que necesariamente hubieran sido consecutivas.
Si existen varias líneas DATA, en diferentes lugares del programa, estas se leen consecutivamente, como si se tratara de una única línea.

Ejemplos

```
10 READ A: READ B
20 DATA 3, 5
```

A las variables A y B se les asignan respectivamente los valores 3 y 5.

```
10 READ A
20 READ B: READ C
30 DATA 4. 7
```

Esta vez falla el programa: a la variable C no le corresponde ningún valor, y provocaremos el mensaje OUT OF DATA.

```
10 READ A
20 READ A$
30 READ C
40 DATA 3, "4", 5
```

A, A$ y C toman respectivamente los valores 3, "4" y 5. Nota las comillas en el 4: la segunda variable es de tipo cadena. El 4 se insertará en la memoria como carácter y no como número.

```
10 READ C$, A, D$
20 DATA "GOOFY", "PLUTO"
```

En esta ocasión los errores son dos: la variable A, siendo de tipo numérico, no puede tomar el valor PLUTO (aparecerá el mensaje de error: NONSENSE IN BASIC) y la variable D$ carece de su correspondiente DATA.

```
10 FOR I = 1 TO 6
20 READ K$
30 PRINT K$
40 NEXT I
50 DATA "FRANCISCO", "MARIO"
60 DATA "MATEOS", "CARLOS"
70 DATA "PEPE", "SANTIAGO"
```

La variable K$ irá tomando sucesivamente los 6 valores especificados en las instrucciones DATA. La instrucción PRINT K$, situada dentro del ciclo FOR, provocará la impresión de FRANCISCO, MARIO,... SANTIAGO.
Observa como la variable K$ es del mismo tipo (cadena) que los valores que deberá tomar como consecuencia de las READ.

```
 5 CLS
10 PRINT "SISTEMA SOLAR" : PRINT
15 FOR K = 1 TO 9
20 PRINT
25 READ P$, D
30 PRINT "EL PLANETA"; P$
35 PRINT "TIENE UN DIAMETRO DE"; D; "KM"
40 NEXT K
45 DATA "MERCURIO", 4880, "VENUS", 12096
50 DATA "TIERRA", 12740, "MARTE", 6780
55 DATA "JUPITER", 141560, "SATURNO", 120800
60 DATA "URANO", 51000, "NEPTUNO", 49300
65 DATA "PLUTON", 7000
```

Esto, más que un ejemplo, es una demostración de la utilidad de READ... DATA. Con pocas instrucciones resulta posible insertar permanentemente informaciones que nunca van a cambiar de una a otra ejecución, permitiendo un ahorro notable de tiempo y espacio.

Sintaxis de la instrucción DATA

DATA constante [, constante] [, ...]

Sintaxis de la instrucción READ

READ variable [, variable] [, ...]

RESTORE

Para mantener bajo control el orden en el que son tomados los valores de las DATA, podemos imaginarnos que en el interior de la memoria existe una especie de índice (llamado puntero) que va marcando cada vez cuál debe ser el valor sucesivo que puede ser leído con una instrucción READ. Cada vez que el ordenador lee un elemento cualquiera desde las líneas DATA, este puntero es desplazado para que indique el elemento sucesivo, de forma que el intérprete BASIC siempre sepa hasta



donde ha llegado la lectura de las DATA. A cada READ, por lo tanto, le corresponderá un avance automático del puntero de las DATA.

Sin embargo, a veces puede resultar muy útil tener la posibilidad, durante el transcurso de un programa, de acceder a las mismas informaciones



más de una vez.
La instrucción RESTORE sirve precisamente para devolver el puntero a la primera línea DATA del programa, haciendo que las constantes estén de nuevo a disposición, como si nunca hubieran sido leídas.
Cuando se encuentra la instrucción RESTORE –y con independencia del número de elementos que ya hayan sido leídos– se provoca el que la siguiente instrucción READ vuelva atrás para leer el primer elemento de la lista de las DATA, igual que si fuera la primera vez.
Así, el siguiente programa.

```
10 FOR I = 1 TO 1000
20 READ A
25 PRINT A
30 RESTORE
40 NEXT I
50 DATA 27, 10, 60
```

tendrá como único resultado el que se imprima 1000 veces el valor 27, dado que el RESTORE situado en la línea 30 le impide al puntero de los DATA avanzar hasta el término siguiente.
Ahora prueba a quitar la línea 30 y a poner en marcha el programa: si tenías dudas sobre el funcionamiento de RESTORE, desaparecerán en seguida.
Si se desea, RESTORE puede ser seguido por un número de línea. En este caso el puntero es desplazado, en vez de al primer elemento de la primera DATA, al primer elemento de la DATA correspondiente al número especificado.

Sintaxis de la instrucción

RESTORE [número de líneas]

Calendario

El primer ejemplo de nuestra lección es una útil aplicación sobre el uso de las instrucciones READ/DATA. Como resultado de la ejecución de este programa se obtiene la impresión de un calendario de cualquier año a partir de 1984. Veamos de qué manera. En primer lugar es necesario conocer qué día de la semana corresponde al primer día del año escogido. Esto se puede lograr de una forma muy sencilla, teniendo bien presente una fecha en concreto: por ejemplo el 1 de Enero de 1984 era Domingo.
A partir de aquí es suficiente con calcular los días que transcurren desde el principio del



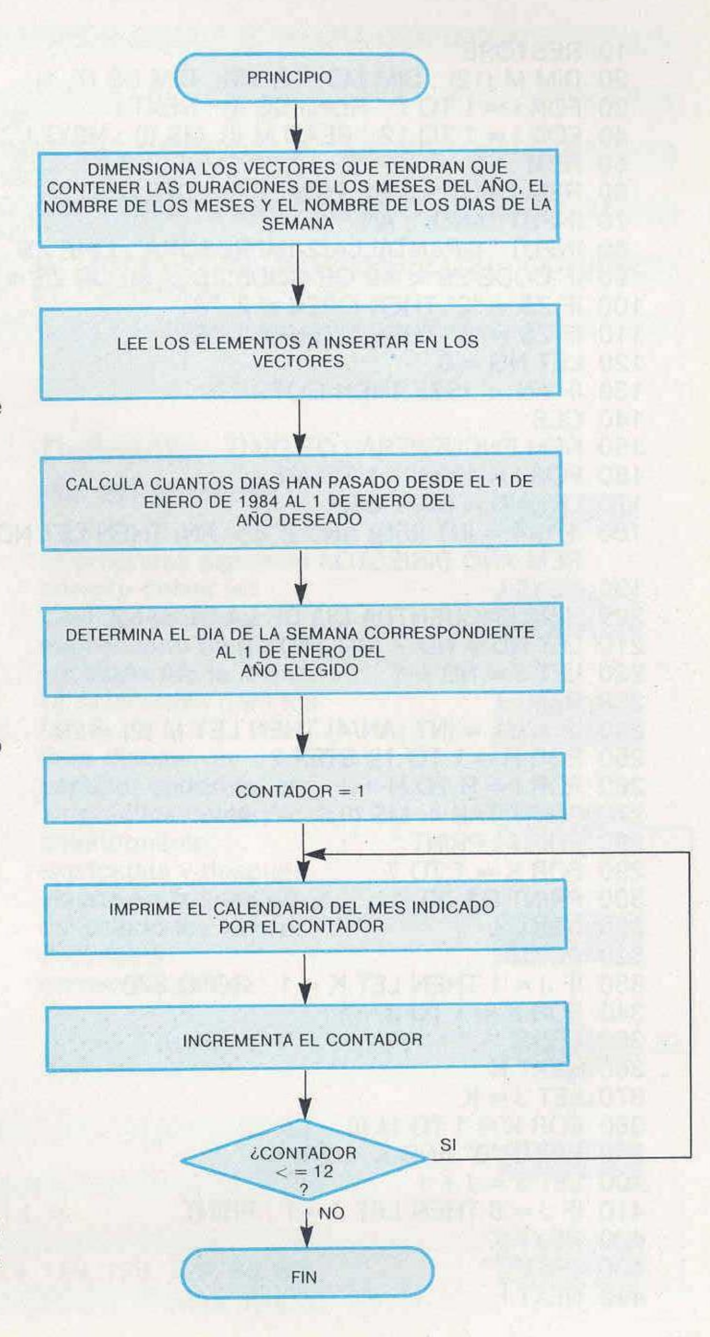
año elegido y el principio de 1984 (y teniendo en cuenta como es natural los años bisiestos) para saber automáticamente en qué día de la semana cae el primero de Enero del año elegido.
Tomemos, por ejemplo, el 1986. Entre el 1 de Enero de 1984 y el 1 de Enero de 1986 hay 731 días (365 + 366 puesto que el 84 era bisiesto). Ahora será suficiente con dividir 731 por 7

731 : 7 = 104 con un resto de 3

para saber que el 1 de Enero de 1986 será un Miércoles. El 3 del resto nos dice precisamente que el principio del 86 cae tres días después que el principio del 84: por lo tanto (y con una escritura quizá no muy correcta)

DOMINGO + 3
= MIERCOLES.

A partir de ahora el camino es bien fácil: bastará con imprimir uno detrás del otro los meses del año, cuidando que la puesta en página sea correcta, para obtener el calendario.

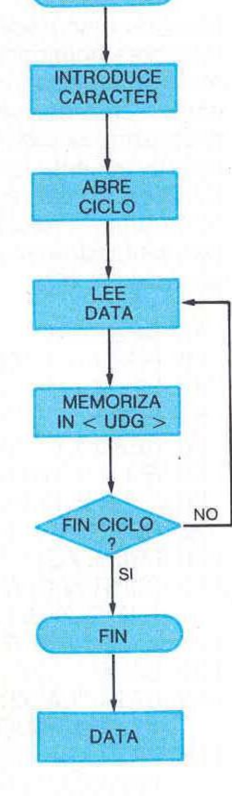


```
 10 RESTORE
 20 DIM M (12) : DIM M$ (12, 10) : DIM D$ (7, 1)
 30 FOR I = 1 TO 7 : READ D$ (I) : NEXT I
 40 FOR I = 1 TO 12 : READ M (I), M$ (I) : NEXT I
 50 REM
 60 REM CALCULA EL PRIMER DIA DEL AÑO
 70 INPUT "AÑO "; AN
 80 INPUT "1-PANTALLA/2-IMPRESORA"; LINE Z$
 90 IF CODE Z$ < 49 OR CODE Z$ > 50 OR Z$ = " " THEN GOTO 80
100 IF Z$ = "2" THEN OPEN # 2, "P"
110 IF Z$ = "1" THEN OPEN # 2, "S"
120 LET NG = 0
130 IF AN < 1985 THEN GOTO 70
140 CLS
150 REM ENCUENTRA LOS DIAS
160 FOR I = 1984 + 1 TO AN
170 LET ND = ND + 365
180 IF (I/4 = INT (I/4)) AND (I <> AN) THEN LET ND = ND + 1:
    REM AÑO BISIESTO
190 NEXT I
200 REM ENCUENTRA DIA DE LA SEMANA
210 LET ND = ND - (INT (ND/7)) * 7
220 LET J = ND + 1
230 REM
240 IF AN/4 = INT (AN/4) THEN LET M (2) = 29
250 FOR H = 1 TO 12 STEP 2
260 FOR I = H TO H + 1
270 PRINT TAB 4; M$ (I); "    "; AN
280 PRINT : PRINT "  ";
290 FOR K = 1 TO 7
300 PRINT G$ (K); "     ";
310 NEXT K
320 PRINT
330 IF J = 1 THEN LET K = 1 : GOTO 370
340 FOR K = 1 TO J - 1
350 PRINT "         ";
360 NEXT K
370 LET J = K
380 FOR K = 1 TO M (I)
390 PRINT "0" AND K < 10; K; "          ";
400 LET J = J + 1
410 IF J = 8 THEN LET J = 1 : PRINT
420 NEXT K
430 PRINT '''
440 NEXT I
```

```
450 IF Z$ = "1" THEN PRINT # 0; AT 1, 8; "PULSA UNA TECLA" :
    PAUSE 0 : CLS
460 NEXT H
470 CLOSE # 2 : STOP
480 DATA "LUNES", "MARTES", "MIERCOLES", "JUEVES", "VIERNES",
  "SABADO", "DOMINGO"
490 DATA 31, "ENERO", 28, "FEBRERO", 31, "MARZO", 30, "ABRIL", 31,
    "MAYO", 30, "JUNIO"
500 DATA 31, "JULIO", 31, "AGOSTO", 30, "SEPTIEMBRE", 31, "OCTUBRE",
    30, "NOVIEMBRE", 31, "DICIEMBRE"
```

Definición de un carácter

El programa siguiente te permite definir un carácter gráfico y memorizarlo en una de las letras (de la A a la U) reservadas para los U.D.G.
Para diseñar un carácter concreto, pon en práctica las técnicas anteriormente explicadas y después inserta en la línea 60 del listado los valores decimales correspondientes.




```
10 INPUT C$
20 FOR I = 0 TO 7
30 READ N
40 POKE USR C$ + I, N
50 NEXT I
60 DATA 60, 66, 129, 129, 129, 129, 66, 60
```

EJERCICIOS

¿Qué palabras (constantes de cadena) serán impresas?

```
 10 READ A : READ B : RESTORE A * B : READ A$
 20 PRINT A$
 30 STOP
 90 DATA 10, 13
100 DATA "VIDEO"
110 DATA "BASIC"
120 DATA "SOFTIDEA"
130 DATA "JACKSON"
140 DATA "SINCLAIR"
150 DATA "SPECTRUM"
160 DATA "16 K"
170 DATA "48 K"
180 DATA "PLUS"
190 DATA "QUANTUM LEAP"
```

Muchos discursos de hombres políticos parecen tener una estructura establecida por este programa; si cambiáramos oportunamente las líneas DATA por palabras como "coyuntura", "plataforma", etc., obtendríamos un claro ejemplo de ello.

```
 10 RESTORE
 20 FOR I = 1 TO 2
 40 LET A = INT (RND * 10)
 50 FOR K = 1 TO A
 60 READ Z$ : NEXT K
 70 IF I = 1 THEN READ A$
 80 IF I = 2 THEN READ B$
 90 RESTORE 150 : NEXT I
100 PRINT A$; " "; B$
110 PRINT # 0; AT 1, 0; "¿OTRA MAS? (S/N)" : LET Z$ = INKEY$ : IF Z$ = " "
    THE GOTO 110
120 IF Z$ = "S" THEN RUN
130 STOP
140 DATA "LA JIRAFA", "EL HOMBRE", "LA VACA", "EL ASNO", "EL CABALLO",
    "EL ELEFANTE", "EL PERRO", "EL GATO", "LA GALLINA", "EL CONEJO"
150 DATA "SE RIE COMO UNA HIENA", "EXPLOTA", "DA SALTITOS",
    "REVOLOTEA", "BOTA", "HUYE", "RELINCHA", "BARRITA", "MUGE",
    "SONRIE"
```

ESTE ES EL SIMBOLO DE COMMODORE. COMPAÑIA AMERICANA. LIDER MUNDIAL EN NUMERO DE ORDENADORES INSTALADOS.

commodore

1541

Su Commodore 64 tiene mucho que decirle.

Unidad de Disco.

El Commodore 64 es el resultado de la experiencia internacional de Commodore como líder indiscutible en el mercado de los microordenadores.

El Commodore 64 es el ordenador más completo y potente de su categoría,... pero todavía tiene mucho que decirle.

Por ejemplo su Unidad de Disco.

Sienta como aumenta notablemente la capacidad de memoria de su C-64, como agiliza la carga y descarga de programas y facilita la localización, casi instantánea, de cualquier dato.

Amplie las posibilidades de su C-64, descubriendo su extensa gama de periféricos.

Ahora que ya sabe que su Commodore 64 tiene todavía mucho que decirle, prepárese a conocerle mejor.

PRINCIPALES CARACTERISTICAS

– 170 K de capacidad – Ficheros secuenciales y relativos y de acceso directo – Unidad inteligente, con sistema operativo incorporada.





Microelectrónica y Control

c/ Valencia, 49-53 08015 Barcelona - c/ Princesa, 47 3.º G 28008 Madrid

Unico representante de Commodore en España.